1. Easy: Understanding Bubble Sort Manually sort the array ` [5,3,8,7] ` using the bubble sort technique.

write-down each step of the process.

a)

given array is [5, 3 , 8 , 7]

lets apply Bubble sort algorithm

now the index of given array is starts from 0 to 3 because the size of array is 4 .

step1 :

> Compare 5 and 3, swap because here "5" is greater than "3": [3, 5, 8, 7]

> Compare 5 and 8, no swap: [3, 5, 8, 7]

> Compare 8 and 7, swap because here "8" is greater than "7" : [3, 5, 7, 8]

after one loop now the array is [3, 5, 7, 8]

step2 :

> Compare 3 and 5, no swap: [3, 5, 7, 8]

> Compare 5 and 7, no swap: [3, 5, 7, 8]

after second loop now the given array is [3, 5, 7, 8]

step3:

> Compare 3 and 5, no swap: [3, 5, 7, 8]

> Compare 5 and 7, no swap: [3, 5, 7, 8]

> Compare 7 and 8, no swap: [3, 5, 7, 8]

∴ After applying Bubble Sort algorithm the array is [3, 5, 7, 8] .

2. Intermediate: Trace the Bubble Sort Provided the unsorted array ` [ 15, 36, 12, 28, 18] ` and trace the bubble sort algorithm step by step,

showing the changes in the array after each pass.

A)

given array is [15, 36, 12, 28, 18]

lets apply Bubble sort algorithm

now the index of given array is starts from 0 to 4 because the size of array is 5 .

step1 :

> Compare 15 and 36, no swap: [15, 36, 12, 28, 18]

> Compare 36 and 12, swap because is 36 greater than 12 : [15, 12, 36, 28, 18]

> Compare 36 and 28, swap because 36 is greater than 28: [15, 12, 28, 36, 18]

> Compare 36 and 18, swap because 36 is greater than 18: [15, 12, 28, 18, 36]

after one loop now the array [15, 12, 28, 18, 36]

step2:

> Compare 15 and 12, swap because 15 is greater than 12: [12, 15, 28, 18, 36]

> Compare 15 and 28, no swap: [12, 15, 28, 18, 36]

> Compare 28 and 18, swap because 28 is greater than 18: [12, 15, 18, 28, 36]

after second loop now the given array is [12, 15, 18, 28, 36]

step3:

>Compare 12 and 15, no swap: [12, 15, 18, 28, 36]

> Compare 15 and 18, no swap: [12, 15, 18, 28, 36]

> Compare 18 and 28, no swap: [12, 15, 18, 28, 36]

after again looping now the given array is [12, 15, 18, 28, 36]

step4:

> Compare 12 and 15, no swap: [12, 15, 18, 28, 36]

> Compare 15 and 18, no swap: [12, 15, 18, 28, 36]

> Compare 18 and 28, no swap: [12, 15, 18, 28, 36]

> Compare 28 and 36, no swap: [12, 15, 18, 28, 36]

∴ After applying Bubble Sort algorithm the array is [12, 15, 18, 28, 36] .

3. Intermediate: Code Implementation Implement the bubble sort algorithm in C++.

Provide them with the following unsorted array: [4, 2, 7, 3, 0] .

Code from scratch and test it to ensure it works correctly

A)

the given array is [4,2,7,3,0]

Code in C++ :

```
#include <iostream>
using namespace std;

int main()
 {
   int arr[] = {4,2,7,3,0};
   int n = sizeof(arr) / sizeof(arr[0]);
   for(int i = 0; i < n-1; i++)
    {
      for (int j = 0; j < n-i-1; j++)
         {

           if (arr[j] > arr[j+1])
            {
             int temp = arr[j];
             arr[j] = arr[j+1];
             arr[j+1] = temp;
             }
          }
     }
   cout << "Sorted array: \n";
   for(int i = 0; i < n; i++)
  cout << arr[i] << " ";
   return 0;
  }
```

∴ The output of the given array is [0, 2, 3, 4, 7].

4.Advanced: Optimization Challenge Challenge yourself to optimize the bubble sort algorithm.

Provided with the partially sorted array ` [11, 22, 33, 44, 55, 100, 99, 88, 77, 66] `.

Optimize the algorithm to reduce the number of comparisons or swaps, making the sorting process more efficient.

A).

Given array [11, 22, 33, 44, 55, 100, 99, 88, 77, 66]

Code in C++ :

```
#include <iostream>
using namespace std;


int main()
 {
   int arr[] = {1, 2, 3, 4, 5, 10, 9, 8, 7, 6};
   int n = sizeof(arr) / sizeof(arr[0]);
   for(int i = 0; i < n-1; i++)
    {
      for (int j = 0; j < n-i-1; j++)
          {


           if (arr[j] > arr[j+1])
            {
             int temp = arr[j];
             arr[j] = arr[j+1];
             arr[j+1] = temp;
             }
          }
     }
   cout << "Sorted array: \n";
   for(int i = 0; i < n; i++)
```

```
          cout << arr[i] << " ";
           return 0;
          }
```

∴ The output of the given array is [11, 22, 33, 44, 55, 66, 77, 88, 99, 100].

5. Advanced: Comparison with Other Sorting Algorithms Compare the bubble sort algorithm with quicksort and merge sort.

Discuss the advantages and disadvantages of bubble sort in different scenarios.

Additionally, analyze when it might be preferable to use other sorting algorithms.

A).

Bubble sort , quick sort and merge sort all are sorting algorithm but different in the properties .

Bubble Sort is simple but less efficient

Quick Sort is efficient but can have a worst-case scenario

Merge Sort is efficient and stable

Bubble sort :

> Bubble Sort has a time complexity of O(n^2), which means it can be inefficient for large datasets

> It repeats this process until the list is sorted.

Quick sort :

> Quick Sort is a divide-and-conquer algorithm.

> It selects a pivot element and partitions the array into two sub-arrays,

one with elements smaller than the pivot and the other with elements larger than the pivot .

Merge sort :

> Merge Sort is also a divide-and-conquer algorithm.

> It divides the array into two halves, recursively sorts each half, and then merges them back together